Chapitre 3

Méthodes arborescentes

Il s'agit d'une famille d'algorithmes qui énumèrent de manière implicite l'ensemble des solutions d'un problème. Leur objectif est de trouver la solution optimale exacte.

3.1 Introduction

Considérons un problème de maximisation :

$$\begin{cases} Max & f(X) \\ X \in & \mathcal{D} \end{cases}$$

Une méthode arborescente consiste à faire évoluer dans le temps une arborescence selon certaines règles.

A chaque noeud de l'arborescence est associé un sous-ensemble de solutions réalisables (c'est à dire une partie du domaine D).

Pour un noeud S on note $D(S)$ l'ensemble des solutions associé.

Définition 3.1 **Arborescence valide**
Une arborescence est dite **valide** si elle vérifie la propriété suivante :

- La racine vérifie $D(R) = \mathcal{D}$
- Si un noeud S a k fils $S_1, \ldots, S_k$, alors

$$D(S) \subset \cup_{i=1}^{k} D(S_i)$$

On a en fait le plus souvent l'égalité.

Considérons un problème de sac à dos à 5 objets défini par :

$P = 6$

i	1	2	3	4	5	6
w_i	8	10	2	5	3	1
p_i	1	2	1	3	2	1

l'arborescence de quatre noeuds suivante est une arborescence valide pour le problème de sac à dos.

- La racine S_0 a deux fils S_1 et S_2.
- Le noeud S_1 a deux fils S_3 et S_4.
- $D(S_1)$ est l'ensemble des sacs de poids ≤ 6 contenant l'objet 1.
- $D(S_2)$ est l'ensemble des sacs de poids ≤ 6 ne contenant pas l'objet 1.
- $D(S_3)$ est l'ensemble des sacs valides contenant l'objet 1 et l'objet 2.
- $D(S_4)$ est l'ensemble des sacs valides contenant l'objet 1 et pas l'objet 2.

La constitution d'une arborescence dépend du problème d'optimisation. Et même il peut y avoir différentes manières de définir les arborescences valides pour un même problème.

Définition 3.2 **Opération de séparation**
L'une des règles d'évolution d'une arborescence est dite **opération de séparation**. Elle consiste à choisir une feuille S d'une arborescence valide dite **ouverte** (on précisera ensuite le sens de cela) et à lui créer des fils pour constituer une nouvelle arborescence valide.

Par exemple, pour l'arborescence ci-dessus, on peut choisir S_3 et lui donner deux fils S_5 et S_6 définis par :

- $D(S_5)$ contient tous les sacs avec les objets $1, 2, 4$.
- $D(S_6)$ contient tous les sacs avec les objets $1, 2$ et sans l'objet 4.

Dans certains cas, on peut constater qu'on ne peut plus séparer un noeud. Par exemple pour le problème de sac à dos, on peut se douter que lorsqu'un noeud est à une profondeur n s'il y a n objets, on a pris des décisions pour tous. Mais on peut parfois s'en appercevoir avant. Par exemple, ici, pour le noeud S_5, on voit que les objets $1, 2, 4$ remplissent entièrement le sac. Donc $D(S_5)$ ne contiendra pas d'autre sac réalisable que le sac comportant les objets $1, 2, 4$. On dit alors que le noeud est **terminal**

Définition 3.3 **Noeud terminal**
Un noeud S d'une arborescence est dit terminal lorsque $|D(S)| = 1$.

L'autre aspect important d'une méthode arborescente est de maintenir

une variable globale, que nous noterons X^* qui mémorise la meilleure solution du problème rencontrée par la méthode.

Définition 3.4 **Opération de mise à jour**
Lorsqu'une arborescence contient un noeud terminal S, il lui correspond donc une seule solution réalisable X puisque $\{X\} = D(S)$. Si $F(X) > F(X^*)$ ou que X^* n'est pas encore défini, on met à jour $X^* \leftarrow X$. De plus, dans tous les cas, le noeud S est dit **fermé**, et ne peut pas être séparé.

Mais si l'on développe l'arborescence en utilisant uniquement les opérations de séparation et de mise à jour, on ne trouvera pas la solution optimale avant d'avoir exploré tous les noeuds terminaux. Ou plus exactement, on ne pourra prouver l'optimalité d'une solution avant d'avoir exploré ces noeuds.

Dans le sac à dos, il peut y avoir jusqu'à 2^n noeuds terminaux.

L'intérêt d'une méthode arborescente va reposer sur sa capacité à ne pas développer tous les noeuds terminaux, mais à tronquer la recherche. L'opération de troncature va reposer sur la définition suivante :

Définition 3.5 **Fonction d'évaluation par excès**
Soit A une arborescence valide pour un problème de maximisation. On appelle fonction d'évaluation par excès une fonction qui à chaque noeud S de A associe une valeur $h(S)$ telle que :

$$h(S) \geq \max_{X \in D(S)} F(X)$$

Exercice 3.1 Evaluation pour le sac à dos

1 Trouver comment définir une fonction d'évaluation par excès pour les arborescences valides définies pour le problème du sac à dos.

2 L'appliquer à l'arborescence précédemment construite.

Solution

Les étudiants donnent en général comme évaluation la somme des profits de tous les objets, sauf ceux qu'on n'a pas mis dans le sac.

Supposons maintenant qu'une arborescence valide comporte un noeud S vérifiant la propriété suivante :

$$F(X^*) \geq h(S)$$

On déduit de cette inégalité qu'aucune solution de $D(S)$ n'aura une valeur d'objectif meilleure que celle de la meilleure solution rencontrée X^*. Il est

donc inutile de chercher la solution optimale du problème dans l'ensemble $D(S)$. On procède alors à :

Définition 3.1 **Opération de troncature**
Lorsque pour un noeud S,$F(X^*) \geq h(S)$, le noeud S est dit **fermé**.

Exercice 3.2 Etat d'une arborescence

Considérons une arborescence valide pour un problème d'optimisation définie comme suit :

- la racine S_0 possède trois fils S_1, S_2, S_3.
- S_2 possède deux fils S_4, S_5.
- On suppose que S_4 et S_3 sont des noeuds terminaux, et que les valeurs des solutions associées sont 455 et 432.
- On a $h(S_0) = 480$, $h(S_1) = 455$, $h(S_5) = 450$, et $h(S_2) = 470$

1 Que vaut $F(X^*)$?

2 Quelles opérations peut-on effectuer sur cette arborescence ?

3 Que peut-on déduire de l'état de l'arborescence après ces opérations ?

Une méthode arborescente consiste à définir, par rapport à un problème d'optimisation donné, au minimum :

- Comment est constituée une arborescence valide
- Comment est réalisée une opération de séparation
- Comment est définie l'évaluation par excès.

La méthode elle même consiste à définir pour arborescence initiale un seul noeud ouvert(la racine), et à effectuer, tant qu'il reste un noeud ouvert, une opération de séparation, ou de mise à jour, ou de troncature.

Lorsque la boucle s'arrête tous les noeuds sont fermés et X^* est la solution optimale.

Quelques éléments de preuve : A l'arrêt de la boucle, une solution optimale $\hat{X}$ se trouve dans l'un des domaines associés aux noeuds feuilles. Si ce noeud S est terminal, c'est que X^* a pris pour valeur $\hat{X}$ lorsque la dernière mise à jour à été effectuée. Sinon, par définition de h, $h(S) \geq F(\hat{X})$. Donc le noeud n'a pu être fermé par troncature que si $F(X^*) \geq F(\hat{X})$. X^* est donc une solution optimale (il peut y en avoir plusieurs.

3.2 Trouver une bonne fonction d'évaluation

Une fonction d'évaluation par excès doit permettre de tronquer les noeuds le plus possible. Il faut donc qu'elle reflète le plus possible la réalité, c'est à dire que la valeur de $h(S)$ soit proche de la valeur qu'elle majore : $\max_{X \in D(S)} F(X)$.

Pour construire une fonction d'évaluation on a souvent recours à **la relaxation de contraintes**.

Considérons le problème de maximisation associé à un noeud S

$$\begin{cases} Max & F(X) \\ X \in & D(S) \end{cases}$$

Supposons que l'on puisse définir $D'(S)$ tel que $D(S) \subset D'(S)$ et que le problème d'optimisation

$$\begin{cases} Max & F(X) \\ X \in & D'(S) \end{cases}$$

soit facile à résoudre. Alors on pose

$$h(S) = \max_{X \in D'(S)} F(X)$$

Par exemple, si le problème d'optimisation associé à un noeud S est un programme linéaire en nombres entiers on peut appliquer la **relaxation continue** qui consiste à considérer que les variables sont à valeurs réelles (et non plus entières).

Pour le sac à dos, à un noeud S correspond un sous-ensemble d'objets choisis, noté $C(S)$, un sous-ensemble d'objets rejetés, noté $R(S)$, les objets restants sont notés $N(S)$. Le problème d'optimisation correspondant s'écrit :

$$\begin{cases} Max & \sum_{i \in N(S)} w_i x_i + \sum_{i \in C(S)} w_i \\ & \sum_{i \in N(S)} p_i x_i \leq P - \sum_{i \in C(S)} p_i \\ \forall i \in N(S), & x_i \in \{0, 1\} \end{cases}$$

Sa relaxation continue s'écrit :

$$\begin{cases} Max & \sum_{i \in N(S)} w_i x_i + \sum_{i \in C(S)} w_i \\ & \sum_{i \in N(S)} p_i x_i \leq P - \sum_{i \in C(S)} p_i \\ \forall i \in N(S), & 0 \leq x_i \leq 1 \end{cases}$$

Pour le sac à dos, on a en outre une propriété très intéressante de la relaxation continue. En effet, si l'on considère que les objets sont numérotés par ordre des $\frac{w_i}{p_i}$ décroissants, alors la solution de la relaxation continue du problème de sac à dos à n objets est donnée par l'algorithme suivant :

Trouver le premier indice k tel que $\sum_{i=1}^{k} p_i \leq P$ et $\sum_{i=1}^{k+1} p_i > P$.

La solution de la relaxation continue est donnée par : $x_i = 1$ pour $i < k$, $x_i = 0$ pour $i > k$, et

$$x_k = \frac{P - \sum_{i=1}^{k-1} p_i}{p_k}$$

Exercice 3.3 Solution continue du sac à dos

1 Calculer la solution continue pour le problème de sac à dos considéré plus haut.

2 Développer une méthode arborescente pour ce problème en utilisant la fonction d'évaluation par excès basée sur la relaxation continue.

3 Peut-on utiliser la fonction d'évaluation pour mieux choisir la variable de séparation ?

4 On peut utiliser les caractéristiques de l'évaluation par excès pour calculer en chaque noeud une solution heuristique. Comment peut-on utiliser cette solution pour fermer des noeuds ?

5 Comment peut-on choisir le noeud à séparer, lorsqu'il y en a plusieurs ?

3.3 Modes de parcours et évaluation par défaut

Pour un problème donné, une méthode arborescente va également préciser certains points :

3.3.1 Stratégies de parcours

- Le choix du noeud ouvert à séparer, lorsqu'il y en a plusieurs
- Le choix de la séparation du noeud, lorsque plusieurs sont possibles.

Les règles qui permettent de choisir sont de nature empirique. Le but est de constituer une arborescence valide dont le plus possible de noeuds seront tronqués.

On distingue, sur le premier point, plusieurs stratégies de choix du noeud à séparer, dites parcours de l'arborescence.

Définition 3.1 **Parcours en largeur d'abord**
Le noeud à séparer est toujours celui qui a été créé en premier (les noeuds ouverts sont gérés avec une file. On emploie cette stratégie lorsque l'on a les moyens d'avoir dès le départ une bonne solution X^* et que l'on espère pouvoir élaguer l'arbre rapidement.

Définition 3.2 **Parcours en profondeur d'abord**
Le noeud à séparer est celui qui a été créé en dernier (les noeuds ouverts sont gérés avec une pile). On emploie ce parcours lorsque l'on ne dispose pas de solution réalisable au départ, et que l'on a besoin d'en avoir une rapidement pour pouvoir commencer à élaguer. Ainsi, cette stratégie explore au départ un seul chemin jusqu'à une feuille.

Définition 3.3 **Parcours du meilleur d'abord**
Le noeud à séparer est celui qui semble le plus prometteur, c'est à dire celui pour lequel $h(S)$ est maximal. On emploie ce parcours lorsque la fonction d'évaluation est très proche de la valeur optimale qu'elle borne, et que le problème n'est pas trop contraint. On atteint ainsi rapidement la solution optimale, même s'il est long de prouver l'optimalité (le temps que la méthode arborescente se termine).

En pratique, on peut combiner les différentes stratégies, comme par exemple, d'abord effectuer une stratégie en profondeur d'abord, puis meilleur d'abord puis largeur d'abord pour finir.

Il est toujours nécessaire, lorsque l'on développe une méthode, de l'expérimenter sur des données réelles et de voir comment elle se comporte (il y a parfois des surprises).

Enfin, il est également possible (et parfois souhaitable) de requérir l'aide du hasard, en tirant aléatoirement le noeud que l'on choisit.

3.3.2 Evaluation par défaut

Afin d'accélérer la méthode, on peut aussi avoir recours à l'évaluation par défaut.

Définition 3.4 **Evaluation par défaut**
Une fonction d'évaluation par défaut est une fonction g qui à chaque noeud S de l'arborescence associe un nombre tel que :

$$g(S) \leq \max_{X \in D(S)} F(X)$$

Lorsqu'un noeud S vérifie $h(S) = g(S)$, on connait la valeur de la fonction à la solution optimale du sous-ensemble $D(S)$. On peut donc fermer le noeud S (cela s'ajoute aux conditions de troncature), puisqu'en le séparant, on ne trouvera pas de meilleure solution.

Par ailleurs, en général une fonction d'évaluation par défaut s'obtient en construisant une solution de $D(S)$ avec un algorithme heuristique. De ce fait, on peut être amené à modifier X^* si la solution trouvée dans le noeud S est meilleure que X^*.

Dans ce cas l'opération de mise à jour est faite chaque fois que l'on fait appel à l'évaluation par défaut.

Pour le problème du sac à dos, on peut considérer l'heuristique suivante : Soit k l'indice de l'objet frontière dans l'évaluation par excès. On construit une solution en prenant tous les objets jusqu'à $k-1$, et ensuite on cherche parmi les suivants lequel, ou lesquels, rentrent dans le sac et on les y place.

Exercice 3.4 Evaluation par défaut pour le sac à dos

1 Appliquez la méthode arborescente avec l'évaluation par défaut définie ci dessus au problème de sac à dos étudié précédemment.

2 Donner les grandes lignes d'un programme C permettant de résoudre le problème du sac à dos.

3.4 Comment procéder pour un problème de minimisation ?

Lorsque le problème est un problème de minimisation, il suffit dans ce qui précède de changer l'opération de mise à jour (changer X^* lorsque $F(X^*)$ est supérieur à la solution qu'on rencontre), et également d'échanger les rôles de l'évaluation par défaut et de l'évaluation par excès dans la troncature.

Ainsi, on notera $h(S)$ l'évaluation par défaut du noeud S et $g(S)$ son évaluation par excès.

Un noeud est fermé lorsque $h(S)$, son évaluation par défaut est supérieure à $F(X^*)$.

Exercice 3.5 Problème d'ordonnancement sur une machine

Dans la suite de l'exercice, on s'intéresse à la définition d'une méthode arborescente pour résoudre le problème d'ordonnancement suivant :

On a n tâches. Chaque tâche i possède trois paramètres : sa durée p_i, sa date d'échéance d_i et le poids de son retard w_i . Ces tâches doivent être effectuées par une machine, sans temps d'arrêt. Etant donné un ordonnancement, si l'on note C_i la date de fin de la tâche i, le retard de i est noté $T_i = \max(C_i - d_i, 0)$. On cherche un ordonnancement (c'est à dire une permutation σ des tâches) qui minimise $\sum_{i=1}^{n} w_i T_i$.

Dans cet exercice on définit une méthode arborescente pour ce problème.

Un noeud S sera associé à une sous-suite de tâches $J_S = \{i_1, \ldots, i_r\}$, et comprend l'ensemble des ordonnancements σ qui se terminent par la séquence J_S : $\sigma(n) = i_r, \ldots, \sigma(n-r+1) = i_1$. Par exemple, si $J_S = 4, 3$ alors l'ensemble des ordonnancements associés au noeud S est l'ensemble des séquences qui se terminent par $4, 3$. Sur 5 tâches $2, 1, 5, 4, 3$ est l'une de

ces séquences.

Notons $P_S = \sum_{i \notin J_S} p_i$.

1. Montrer que $h(S) = \sum_{s=1}^{r} w_s \max(P_S + \sum_{k=1}^{s} p_{i_k} - d_{i_s}, 0)$ est une évaluation par défaut de la somme des retards au noeud S.
2. Indiquer un algorithme de votre choix permettant de calculer en chaque noeud une évaluation par excès.
3. Un noeud S est séparé en autant de fils qu'il y a de tâches $j \notin J_S$, correspondant aux sous-séquences $j, i_1, \ldots, i_r$. Appliquez la méthode arborescente (au moins partiellement) à l'exemple suivant en précisant, pour les noeuds que vous développerez, le choix du noeud à séparer, les évaluations de chacun des noeuds développés et éventuellement les troncatures utilisées.

i	1	2	3	4	5
p_i	4	3	7	2	5
d_i	5	6	3	8	17
w_i	1	1	1	1	1